# 基于多种群的随机扰动蚁群算法求解分布式约束优化问题 *

石美凤†，肖诗川，冯　欣
(重庆理工大学 计算机科学与工程学院，重庆 400054)

**摘　要：**针对现有的基于蚁群优化思想求解分布式约束优化问题的算法收敛较慢，且容易陷入局部最优等问题，提出了一种基于多种群的随机扰动蚁群算法(Random disturbance based multi-population ant colony algorithm to solve distributed constraint optimization problems，RDMAD)来求解分布式约束优化问题。首先，RDMAD 提出了一种分工合作机制，将种群按比例划分为采用贪婪搜索的子种群和采用启发式搜索的子种群，同时构建分级更新策略，提高算法收敛速度和求解质量；然后，对采用贪婪搜索的子种群设计自适应变异算子和奖惩机制，防止算法陷入局部最优；最后在算法陷入停滞时触发随机扰动策略，增加种群多样性。将 RDMAD 与 7 种最先进的非完备算法在三类基准问题上的寻优结果进行了实验对比，实验结果表明 RDMAD 在求解质量和收敛速度上优势明显，且稳定性较高。

**关键词：**分布式约束优化问题；蚁群算法；自适应变异算子；非完备算法
**中图分类号：**TP18　　doi: 10.19734/j.issn.1001-3695.2022.03.0084

## Random disturbance based multi-population ant colony algorithm to solve distributed constraint optimization problems

Shi Meifeng†, Xiao Shichuan, Feng Xin
(*College of Computer Science & Engineering*, *Chongqing University of Technology*, *Chongqing* 400054, *China*)

**Abstract:** Ant-based algorithm to solve distributed constraint optimization problems (ACO_DCOP) is an excellent population-based algorithm for solving DCOPs. However, ACO_DCOP has some shortcomings including very slow convergence speed and easily falling into local optima. To cope with these issues, this paper proposes a random disturbance based multi-population ant colony algorithm to solve DCOP (RDMAD) . The method introduces a division of labor and cooperation mechanism to divide the population into two subpopulations for greedy search and heuristic search respectively. The method also constructs a hierarchical update strategy to speed up convergence and improve solution quality. Furthermore, this paper designs an adaptive mutation operator and a reward and punishment mechanism for the greedy search subpopulation to prevent RDMAD falling into the local optima. Simultaneously, this paper introduces a random disturbance strategy to increase the population diversity when RDMAD is stagnant. To verify the performance of the proposed algorithm, RDMAD is compared with the other seven advanced incomplete algorithms on three types of benchmark problems. The extensive experimental results show that the proposed algorithm is significantly superior to the state-of-the-arts algorithms in solution quality and convergence speed. In addition, RAMAD is far stable than the competing algorithms.

**Key words:** distributed constraint optimization problems; ant colony algorithm; adaptive mutation operator; incomplete algorithm



## 0　引言

多智能体系统(multi-agent system，MAS)[1]是分布式人工智能领域重要的一部分。分布式约束优化问题(distributed constraint optimization problems, DCOP)[2]是 MAS 基本框架之一，被广泛地应用于许多实际问题建模，如传感器网络[3]、任务调度[4]等。

在过去的二十年里，许多算法被提出用来求解 DCOP。其中完备算法可得到问题的最优解。SyncBB[5]、AFB[6]、ADOPT[7]、BnB-ADOPT[8]等是典型的基于搜索的完备算法。DPOP[9]作为典型的基于推理的完备算法，其利用动态规划思想求解 DCOP，但 DPOP 在求解过程中会遭受指数级的内存消耗。 MB-DPOP[10]算法被提出用于降低 DPOP 算法内存消耗。为提高 MB-DPOP 算法性能，Chen 等[11]提出 RMB-DPOP 算法减少 MB-DPOP 算法中的冗余推理，提高了算法的可扩展性。Rashik 等[12]利用交叉边一致性缩短了 DPOP 的运行时间。由于 DCOP 是 NP-Hard，相比之下，非完备算法虽然不能找到最优解，但在通信和计算方面有更好的性能。其中，基于局部搜索的非完备算法是目前的研究热点，其中包括 DSA[3]和 GDBA[13]等。此外，ALS[14]、PDS[15]和 LSGA[16]等框架被用来提高基于局部搜索的算法的求解质量。Max-Sum[17]和 Max-Sum_ADVP[18]等是基于推理的非完备算法，其中 agent 通过因子图传播和积累效用。基于采样的非完备算法(如 DUCT[19])则通过对搜索空间进行采样来求解 DCOP。近来，出现了一类利用种群求解 DCOP 的非完备算法。Mahmud 等[20]提出了一种利用进化优化思想求解 DCOP 的算法。Chen 等[21]提出了利用蚁群优化思想求解 DCOP 的算法(ACO_DCOP)，ACO_DCOP 是目前唯一利用蚁群优化思想求解 DCOP 的算法，其从传统蚁群算法演变而来，然而 ACO_DCOP 算法中仅利用单种群寻优，同时受信息素影响，收敛较慢，容易陷入局部最优。目前多种群策略被广泛使用，薛宏全等[22]通过分析蚂蚁分工，利用核心蚁群和搜索蚁群的配合有效的解决了车间调度问题。朱佑滔等[23]提出了一种多种群蚁群算法用于求解机械手臂的路径规划问题。陈佳等[24]

**收稿日期：**2022-03-04；**修回日期：**2022-05-03　**基金项目：**重庆市教育委员会科学技术研究计划青年项目资助项目(KJQN202001139)；重庆市基础研究与前沿探索项目(cstc2018jcyjAX0287)；重庆理工大学研究生创新项目(clgycx20203116)；重庆理工大学科研启动基金资助项目(2019ZD03)

**作者简介：**石美凤(1989-)，女(通信作者)，讲师，硕导，博士，主要研究方向为计算智能、分布式人工智能(shimf@cqut.edu.cn)；肖诗川(1996-)，女，硕士研究生，主要研究方向为计算智能、分布式人工智能；冯欣(1982-)，女，副教授，硕导，博士，主要研究方向为计算机视觉.

采用主从蚁群的多种群机制有效求解旅行商问题。因此针对 ACO_DCOP 收敛较慢，且易陷入局部最优等问题，本文提出一种基于多种群的随机扰动蚁群算法求解分布式约束优化问题(RDMAD)。本文的主要贡献包括 4 个方面:

a) 提出了一种分工合作机制，子种群分别通过执行贪婪搜索和启发式搜索使得局部和全局搜索相互协调，可以更好地探索解空间和开发优秀解。

b) 提出了一种随机扰动策略，在算法陷入停滞时，触发随机扰动策略，种群重新划分为 3 个子种群，新增采用随机选值的子种群，增加种群多样性，有助于算法跳出局部最优。

c) 设计了一种分级更新策略，执行不同任务的子种群采用不同更新方式，提高了算法的收敛速度和求解质量。

d) 对 RDMAD 算法的复杂性进行了理论分析，并通过实验结果表明 RDMAD 算法在求解质量和收敛性能上优势明显。

## 1　背景

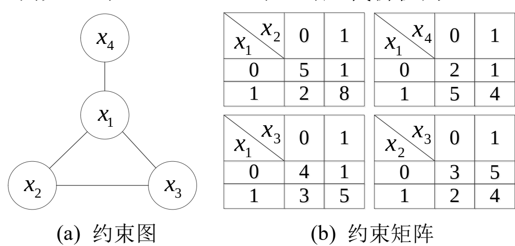
### 1.1　分布式约束优化问题

DCOP 可定义为一个四元组 $\langle X,D,F,A\rangle$ [25]，其中 $A$ 是 agent 的集合 $\{a_1,a_2,\ldots,a_n\}$；$X$ 是离散变量的集合 $\{x_1,x_2,\ldots,x_m\}$，一个 agent 控制一个或多个变量，$m\geq n$；D 是离散变量值域的集合 $\{D_1,D_2,\ldots,D_m\}$，其中 Di 是变量 $x_i$ 的值域；$F$ 是约束关系函数的集合 $\{f_1,f_2,\ldots,f_q\}$，其中 $f_i\in F$ 是子集 $x^i\subseteq X$ 的函数，该函数定义了 $x^i$ 中变量间的约束关系。

DCOP 求解算法的目标为寻找一组赋值组合 $X^*$ 使式(1)所示全局约束代价最小。

$$X^*=\arg\min_X\sum_{i=1}^{q}f_i(x^i) \tag{1}$$

本文中一个 agent 控制一个变量，因此这里“agent”和“变量”可互换。图 1(a)为 DCOP 的约束图，其中一个节点表示一个 agent，两个 agent 之间的边表示它们之间存在约束关系，图 1(b)为 DCOP 的约束矩阵，其中 0、1 为变量的取值，矩阵中其余值为当有约束的两个变量取值时对应的代价大小。例如，当 $x_1$=0，$x_2$=0 时，对应代价值为 5。




| $x_1$ \ $x_2$ | 0 | 1 |
|---|---|---|
| 0 | 5 | 1 |
| 1 | 2 | 8 |

| $x_1$ \ $x_4$ | 0 | 1 |
|---|---|---|
| 0 | 2 | 1 |
| 1 | 5 | 4 |

| $x_1$ \ $x_3$ | 0 | 1 |
|---|---|---|
| 0 | 4 | 1 |
| 1 | 3 | 5 |

| $x_2$ \ $x_3$ | 0 | 1 |
|---|---|---|
| 0 | 3 | 5 |
| 1 | 2 | 4 |

(a) 约束图　　(b) 约束矩阵

图 1　DCOP 实例

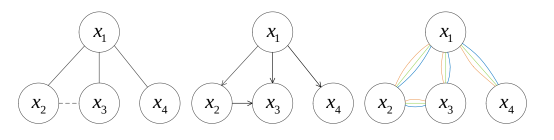
Fig. 1　A DCOP instance

### 1.2　ACO_DCOP

蚁群优化算法(Ant Colony Optimization, ACO)是一种基于种群的求解组合优化问题的元启发式算法，已成功应用于旅行商问题、约束满足问题等。由于 DCOP 中没有实际的路径可用，因此传统的 ACO 无法直接用于求解 DCOP。目前仅有 ACO_DCOP 成功将蚁群优化思想应用于求解 DCOP。

ACO_DCOP 利用 agent 之间的消息传递机制来模拟蚂蚁的运动，首次将群体智能应用于求解 DCOP。以图 1 为例得到图 2(c)所示信息素路径构造图，其中节点表示 agent。 首先将图 1(a)中的约束图转为广度优先搜索伪树[26]，如图 2(a)所示。然后构建的 agent 之间的消息传递顺序(蚂蚁爬行方向)，如图 2(b)所示，其中上层 agent 的优先级高于下层 agent，同层间的 agent 邻居个数越多，值域越大优先级越高，若两个同层的 agent 具有相同的邻居数和值域大小，则 agent 的命名 *id* 越小，优先级越高。 因此 agent $a_i$ 的邻居可分为高优先级邻居 $H_i$ 和低优先级邻居 $L_i$，且消息从高优先级的 agent 传向低优先级的 agent。同时根据 agent 的取值不同，每两个 agent 之间有多条信息素路径。如图 2(c)信息素路径构造图，当 $a_i$ 取值为 $d_i$，$a_j$ 取值为 $d_j$ 时，则该路径上的信息素浓度为 $\tau_{ij}(d_i,d_j)$。

每次迭代中，蚂蚁从优先级最高的 agent 出发，在每只蚂蚁获得取值后，agent 将蚂蚁取值发送给其低优先级邻居。当接收到其高优先级邻居的蚂蚁取值后，agent $a_i$ 首先为每只蚂蚁合并收到的解集合，当 $a_i$ 收到其所有高优先级邻居的蚂蚁取值后，采用转移概率为每只蚂蚁选取变量值域中的取值，否则等待其高优先级邻居的取值。 $a_i$ 完成为每只蚂蚁取值后将蚂蚁取值发送给自己的低或最低优先级邻居。当最低优先级 agent 收到所有蚂蚁取值后，此时每只蚂蚁已经完成解的构建，计算每只蚂蚁构建的解的代价，代价越小解的质量越好，根据代价更新全局最优解并且计算每只蚂蚁的信息素增量，然后将信息素增量发送给所有 agent。agent 收到信息素增量信息后，更新和蒸发与其高优先级邻居间的信息素路径上的浓度，到此一次迭代结束。




(a) BFS 伪树　(b) 消息传递顺序　(c) 构造图

图 2　信息素路径构建

Fig. 2　Pheromone path construction

## 2　RDMAD 算法

本文提出了一种基于多种群的随机扰动蚁群算法用于求解 DCOP，其主要采用分工合作机制、分级更新和随机扰动策略。该算法利用子种群在寻优过程中的不同指导作用，有效地提高了算法收敛和求解性能。

### 2.1　初始化阶段

RDMAD 算法利用蚂蚁在 agent 之间的运动来构造解。首先 RDMAD 算法将 agent 之间的约束图转换为广度优先搜索的伪树结构，然后根据伪树结构构建 agent 消息传递顺序，生成最终构造图，如 1.2 节图 2 所示过程。RDMAD 算法中，信息素信息由低优先级 agent 保存。在完成构造图构建后，初始化参数，并且每个 agent 初始化蚂蚁解集为空。然后优先级最高 agent 为每只蚂蚁随机取值，再将取值信息发送给自己的低优先级邻居。以 1.2 节图 2 为例，$x_1$ 为优先级最高节点，假设种群规模为 2，$x_1$ 为蚂蚁 1 取值为 1，为蚂蚁 2 取值为 0，则将 $x_1$ 的 *id* 和蚂蚁取值{1,0}发送给 $x_1$ 的低优先级邻居 $x_2$ , $x_3$ 和 $x_4$ 。

### 2.2　分工合作机制

基于种群求解 DCOP 的策略近几年才出现，这类算法都是直接从传统群体智能算法中演变而来，仅利用了单种群来寻优。本文在现有采用蚂蚁转移概率取值的种群上增加了采用贪婪搜索的子种群，利用多种群合作更好地平衡开发和探索。

当 agent $a_i$ 接收到来自其高优先级邻居的取值时，首先合并所有取值，当 $a_i$ 接收到所有高优先级邻居的取值时，$a_i$ 开始为蚂蚁选值，每个子种群的选值策略如下：

1) 子种群 1

agent $a_i$ 采用贪婪搜索为子种群 1 中的蚂蚁取值，该方式增强了蚂蚁对局部的探索，有利于提高算法收敛速度。$a_i$ 采用式(2)为蚂蚁 $k$ 选取使 $a_i$ 与其邻居间约束代价和最小时的取值 $d_i$。

$$d_i=\arg\min_{d_i\in D_i}\Big(\sum_{j\in H_i}\cos t_{ij}(d_i,V_{k,j})+est_i(d_i)\Big) \tag{2}$$

其中，$D_i$ 是值域，$V_{k,j}$ 是 $a_i$ 的高优先级邻居 $a_j$ 对蚂蚁 $k$ 的取

值，$\cos t_{ij}(d_j,V_{k,j})$ 是 $a_i$ 为蚂蚁 $k$ 取值为 $d_i$，$a_j$ 为蚂蚁 $k$ 取值为 $V_{k,j}$ 时的约束代价，式(3)定义了 $a_i$ 与其低优先级邻居 $L_i$ 之间的最小约束代价和的预估值 $est_i(d_i)$，其初始化为最优解。

$$est_i(d_i)=\sum_{j\in L_i}\min_{d_j\in D_j}\cos t_{ij}(d_i,d_j) \tag{3}$$

为避免种群因贪婪搜索快速陷入局部最优，本文为子种群 1 设计了一种自适应调整的变异算子 $p_m$，其定义如式(4)所示。

$$p_m=1-m\frac{totalcycle-curcycle}{totalcycle} \tag{4}$$

其中，$m$ 为权值，可调整变异算子大小，$totalcycle$ 为总的迭代次数，$curcycle$ 为当前迭代次数。当 $a_i$ 为种群 1 取值完成时，采用随机概率 $q\in[0,1]$ 挑选蚂蚁个体进行值交换，当小于 $p_m$ 时，$a_i$ 随机选择蚂蚁 $k'$，将当前蚂蚁 $k$ 的取值与蚂蚁 $k'$ 的取值互换，得到新个体。

2) 子种群 2

子种群 2 保留原有的转移概率取值方式，其主要采用蚁群优化思想中的启发式搜索，这有利于蚂蚁对 agent $a_i$ 的解空间进行全局探索。该概率计算依赖于信息素因子和启发式因子，同时配合轮盘赌。$a_i$ 采用式(5)为蚂蚁 $k$ 取值为 $d_i$ 的概率如下。

$$p_{k,i}(d_i)=\frac{\theta_{k,i}(d_i)^{\alpha}\eta_{k,i}(d_i)^{\beta}}{\sum_{d_i'\in D_i}\theta_{k,i}(d_i')^{\alpha}\eta_{k,i}(d_i')^{\beta}} \tag{5}$$

其中，$\alpha$ 和 $\beta$ 分别为信息素因子和启发式因子权重，信息素因子 $\theta_{k,i}(d_i)$ 影响蚂蚁对路径的探索，其定义如式(6)所示。

$$\theta_{k,i}(d_i)=\sum_{j\in H_i}\tau_{ij}(d_i,V_{k,j}) \tag{6}$$

其中，$\theta_{k,i}(d_i)$ 为 $a_i$ 对蚂蚁 $k$ 取值 $d_i$ 时，$a_i$ 与其所有高优先级邻居之间的信息素浓度之和。启发式因子 $\eta_{k,i}(d_i)$ 影响了蚂蚁对解的开发，$a_i$ 与其邻居间的约束代价和越大，启发式因子越小，当前解被开发的可能性越小，如式(7)所示。

$$\eta_{k,i}(d_i)=\frac{1}{\sum_{j\in H_i}\cos t_{ij}(d_i,V_{k,j})+eL_i(d_i)-\mathrm{LC}} \tag{7}$$

其中，LC 为 $a_i$ 与其邻居间的最小约束代价和，用于评估值 $d_i$ 的可开发程度，如式(8)所示。

$$\mathrm{LC}=\min_{d_i\in D_i}\Big(\sum_{j\in H_i}\min_{d_j\in D_j}\cos t_{ij}(d_i,d_j)+eL_i(d_i)\Big)-1 \tag{8}$$

## 2.3 随机扰动策略

种群的多样性影响了算法对解空间的搜索范围，由于蚂蚁是依靠路径上的信息素浓度寻优的，ACO_DCOP 在迭代后期，种群多样性越来越小，因此算法容易陷入局部最优。本文设计了一种随机扰动策略来增加种群多样性。当算法停滞次数等于设定的阈值 $count$，此时触发随机扰动策略。算法将种群重新划分成三个子种群，其中子种群 1 和 2 的任务不变，子种群 3 的任务则是扰动信息素积累，按式(9)采用完全随机取值，打破原有信息素累积规律。

$$d_i=random(D_i) \tag{9}$$

## 2.4 分级更新策略

根据每个子种群的引导作用不同，构建分级更新策略。首先当最低优先级 agent $a_i$ 接收到所有高优先级邻居的取值后，计算每只蚂蚁的信息素增量 $\Delta_k$，如式(10)所示。

$$\Delta_k=1-\frac{\cos t_k-bestcost}{\frac{1}{K}\sum_{k=1}^{K}\cos t_k-bestcost} \tag{10}$$

其中，$\cos t_k$ 为蚂蚁 $k$ 构建的完整值分配对应的代价，$bestcost$ 为全局最优代价值，$K$ 为种群规模。最低优先级 $a_i$ 将信息素增量 $\Delta_k$、种群解集和最优解发送给其他所有 agent。agent 利用信息素增量 $\Delta_k$ 更新与其高优先级邻居之间信息素路径浓度。更新策略如式(11)所示。

$$\tau_{ij}(V_{k,i},V_{k,j})=\tau_{ij}(V_{k,i},V_{k,j})+\Delta_k',\forall j\in H_i \tag{11}$$

$$\Delta_k'=\begin{cases}\delta_{k,ij},k\in\text{子种群1}\\0.8\Delta_k,k\in\text{子种群2}\\1.2\Delta_k,k\in\text{子种群3}\end{cases} \tag{12}$$

式(12)是各子种群信息素增量的定义。由于子种群 1 对种群的探索方向具有引导作用，因此子种群 1 的更新方式按式(13)所示奖惩机制进行。

$$\delta_{k,ij}=\begin{cases}\Delta_k n_1, & \Delta_k\ge 0\\ \Delta_k/n_1, & \Delta_k<0\end{cases} \tag{13}$$

其中，$n_1$ 为子种群 1 的规模。若 $\Delta_k$ 为正，相应路径将获得奖励，否则将受到处罚。该机制可减小路径上信息素浓度的差异，防止算法过快收敛。

分级更新策略体现了多种群在寻优过程中对算法的不同指导作用。通过这种模式，提高了算法的收敛速度和寻优质量。

## 2.5 信息素蒸发

信息素蒸发阶段可使得蚂蚁忘记之前不好的路径，agent 在更新信息素后，根据式(14)进行信息素蒸发，其中 $\rho$ 为蒸发率，$\tau_0$ 为初始化浓度，信息素范围为[ $\tau_{\min}$ , $\tau_{\max}$ ]。

$$\tau_{ij}(d_i,d_j)=(1-r_1\rho)\tau_{ij}(d_i,d_j)+r_2\rho\tau_0 \tag{14}$$

其中，$r_1$，$r_2$ 控制了蒸发率的大小，默认为 1。在触发随机扰动策略时，$r_1$ =2，$r_2$ =0.5，增强信息素蒸发，有助于算法跳出局部最优。

## 2.6 RDMAD 算法步骤

RDMAD 算法具体实现步骤如算法 1 所示。

算法 1　RDMAD 算法(for agent $a_i$ )

输入：初始化参数 $\alpha,\beta,\rho,\tau_0,K,count$ 。

输出：使全局约束代价最小的一组赋值组合 $X^*$。

for each $d_i\in D_i$ do 计算 $est_i(d_i)$ end for
for each 蚂蚁 k do
　初始化解集：$V_{k,*}\leftarrow\{\},V\leftarrow V\cup V_{k,*}$
　if $a_i$ 是优先级最高节点
　　for each 蚂蚁 k do
　　　$a_i$ 为蚂蚁随机取值为 $V_{k,i}$ ，$V_{k,*}\leftarrow V_{k,*}\cup V_{k,i}$
　　end for
　　发送蚂蚁取值信息 $V$ 和 $a_i$ 的 $id$ 给低优先级邻居 $L_i$
end for
When agent $a_i$ 收到了取值信息( $id$ , $recv_V$ )：
　for each 蚂蚁 k do $V_{k,*}\leftarrow V_{k,*}\cup recv_V$ end for
　if $a_i$ 从其高优先级邻居收到了所有取值信息
　　for each 蚂蚁 k do
　　　if c==$count$
　　　　$a_i$ 根据式(2),(4)为子种群 1 取值
　　　　$a_i$ 根据式(5),(9)分别为子种群 2,3 取值
　　　else $a_i$ 根据式(2),(4),(5)为子种群 1,2 取值
　　　合并取值 $V_{k,*}\leftarrow V_{k,*}\cup V_{k,\mathrm{i}}$
　　end for
　　$a_i$ 发送取值信息{ $id$ , $V$ }给 $L_i$ 或最低优先级 $a_i$
　if $a_i$ 优先级最低且收到所有取值信息
　　更新 *bestcost* 和其对应的最佳值分配 $v^*$ ,更新 *count*，按式(10)计算每只蚂蚁的信息素增量 $\Delta_k$
　　发送信息素信息{ $V$ , $\Delta$ , $v^*$ }给所有 agent
When agent $a_i$ 收到了信息素信息( $id$ , $recv_V$ )：
　根据式(11),(14)分别更新和蒸发信息素，更新 $est_i(d_i)$
　if 不满足终止条件 then 开始下一轮循环

在信息素更新和蒸发阶段结束过后，还要对 agent $a_i$ 与其低优先级邻居 $L_i$ 之间的最小约束代价和的预估值 $est_i(d_i)$ 进行

更新。由于在分布式场景中，agent 仅知道其邻居的信息，并且根据消息传递顺序，agent 仅知道其高优先级邻居的取值。因此该预估值可帮助 $a_i$ 估算与其所有邻居的最优代价和，有助于启发式因子对当前取值的评估。 $est_i(d_i)$ 的更新如算法 2 所示。

算法 2　更新预估值 $est_i(d_i)$　(for agent $a_i$ )

输入：种群解集 $V$ 。

输出：更新后的预估值 $est_i(d_i)$ 。

for each　$d_i \in D_i$　do

　sum = 0, num = 0

　for each　蚂蚁 k do

　　if　$V_{k,i} = d_i$ then

　　　$sum += \sum_{j \in L_i} \text{cost}_{ij}(d_i, V_{k,j})$ , num = num+1

　end for

　if num!=0 then

　　ave= sum/num,　$est_i(d_i) = (est_i(d_i) + \text{ave})/2$

end for

## 2.7　RDMAD 算法复杂性分析

RDMAD 算法复杂性分析主要包含算法的消息数、空间复杂度和时间复杂度。将改进后的 RDMAD 算法与 ACO_DOCP 算法进行比较。在每次迭代中，除最低优先级 agent 外，每个 agent 都会向其低优先级邻居(或最低优先级邻居)发送取值信息，最低优先级 agent 会向所有 agent 发送信息素信息。由于取值信息中包含了蚂蚁的解集，因此消息的大小为 $O(nK)$，$n$ 为 agent 个数，因此 RDMAD 算法中值消息大小与 ACO_DCOP 算法一致。包含了蚂蚁的解集、信息素增量和最优解的信息素消息的大小为 $O((K+1)n+K)$，因此 RDMAD 算法中信息素消息大小与 ACO_DCOP 算法一致。

每个 agent　$a_i$ 存储与其高优先级邻居间的信息素路径需要 $O(|H_i||D_i||D_j|)$大小的空间。因此 RDMAD 算法需要的空间大小与 ACO_DCOP 算法一致。时间复杂度主要是取值计算，在最坏情况下，当 $a_i$ 为蚂蚁 $k$ 取值时，对于值域 $D_i$ 中的每个值 $d_i$ 会遍历其所有高优先级邻居的取值，因此 $a_i$ 计算一个值消息需 $O(K|H_i||D_i|)$次操作。因此 RDMAD 算法时间复杂度与 ACO_DCOP 算法一致。

# 3　实验与结果分析

## 3.1　实验设置

实验采用三类基准问题进行算法性能测试，包括随机 DCOPs[27] (EXP-1、EXP-2)、无尺度网络问题[28] (EXP-3、EXP-4)和加权图着色问题 (EXP-5)。具体相关信息如表 1 所示。

表 1　问题配置

Tab. 1　Problem configuration

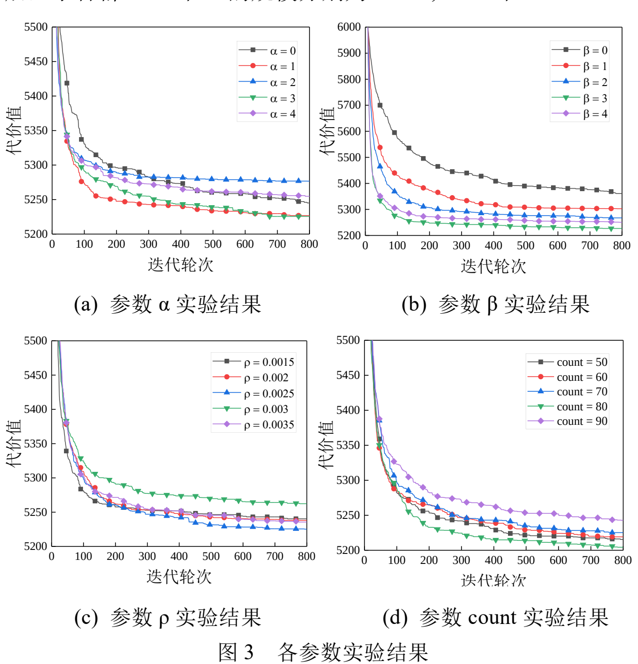
| 测试问题 | agent 个数 | 变量值域 | 约束代价范围 | 问题密度 |
|---|---|---|---|---|
| EXP-1 | 70 | 10 | [1,100] | 0.1 |
| EXP-2 | 70 | 10 | [1,100] | 0.6 |
| EXP-3 | 70 | 10 | [1,100] | 10, 3 |
| EXP-4 | 70 | 10 | [1,100] | 10, 7 |
| EXP-5 | 120 | 3 | [1,100] | 0.05 |

## 3.2　参数分析

为验证重要参数对 RDMAD 算法的影响，以随机 DCOPs (EXP-1)为例，对 RDMAD 算法中的信息素因子 $\alpha$、启发式因子 $\beta$、信息素蒸发率 $\rho$ 以及触发阈值 *count* 进行了实验分析。为保证实验结果的公平性，本实验中采用控制变量法对每个参数进行调节，同时每个参数的每个值取独立运行 30 次的平均值作为该取值下的实验结果。各参数的实验结果如图 3 所示。

图 3(a)显示了 $\alpha$ 变化对 RDMAD 算法性能的影响，从图中可以看到，随着 $\alpha$ 的增大算法求得的解质量也随之提高，但在 $\alpha>1$ 后，算法的性能开始下降，因此根据实验结果，$\alpha$ 取 1 时 RDMAD 算法的性能最好。图 3(b)中的 $\beta$ 的取值也对 RDMAD 算法的性能有较大的影响，从图中可以看出 $\beta$ 越大，算法的求解质量和收敛性能都在提升，但当 $\beta>3$ 后解的质量开始下降。从图 3(c)的实验结果可以得出，信息素的蒸发率 $\rho$ 也会影响算法的性能，但它的影响略小于参数 $\alpha$ 和 $\beta$，当 $\rho=0.0025$ 时，RDMAD 算法得到的解质量最好。最后图 3(d)为触发间隔的选择实验，实验结果表明在触发随机扰动策略时，设置合适的触发间隔能有效提高求解质量。

因此根据实验结果，适当设置重要参数的值有利于提高算法的求解质量和收敛性能。为不失公平性，对比算法的参数采用原文推荐值。RDMAD 算法参数设置为: $\alpha=1, \beta=3, \rho=0.0025, \tau_0=3, count=80, K=20$。在未触发随机扰动策略时，子种群 1、2 的规模分别为 0.5 $K$ , 0.5 $K$ ；在触发随机扰动策略后，子种群 1、2 和 3 的规模分别为 0.5 $K$ , 0.3 $K$ 和 0.2 $K$ 。




(a) 参数 α 实验结果　　(b) 参数 β 实验结果

(c) 参数 ρ 实验结果　　(d) 参数 count 实验结果

图 3　各参数实验结果

Fig. 3　Parametric experimental results

## 3.3　实验结果及分析

为验证 RDMAD 算法的鲁棒性和寻优性能，在测试问题上与其他优秀的非完备 DCOP 算法进行比较，包括 PDS-DSA[15]、GDBA[13]、ACO_DCOP[20]、DSAN[29]、DSA[3]、LSGA_DSA[16]、AED[21]。由于非完备算法具有随机性，因此每个实例取独立运行 30 次的均值为结果，每个测试问题随机生成 20 个实例。

表 2 为 RDMAD 算法与 ACO_DCOP 算法，及其他对比算法在每个测试问题的 20 个实例上的约束代价的均值(Mean)和标准差(Std Dev)。同时，采用 Wilcoxon 符号秩和检验法对实验结果进行统计分析，其中"+"表示 RDMAD 算法在 20 个实例上得到的结果优于对比算法的个数；而"-"则表示相反意思。从表 2 的 Mean 值可以看出，在除 EXP-2 外的所有测试问题上，RDMAD 算法得到的问题的约束代价值小于其他所有对比算法。且在 EXP-2 测试问题上， RDMAD 算法 Mean 值只是略大于 AED 算法。同时，从统计结果来看，RDMAD 算法每次得到的结果都优于 ACO_DCOP、DSA、DSAN 和 GDBA 算法，且在其他测试问题上也具有明显优势，稳定性较高。另外，通过表 2 中的 Wilcoxon 符号秩和检验法的结果 *p-value* 也再一次验证了 RDMAD 算法在求解质量方面优势明显。

图 4 为所有算法在不同测试问题上的收敛曲线。在五个测试问题上，RDMAD 算法均具有优秀的收敛和寻优性能。在 EXP-1 上，RDMAD 算法相比原来的 ACO_DCOP 提高了约 4.2%，相比其他算法提高了约 1.3%～12.9%。从各算法的收敛曲线可以发现，由于缺乏全局信息，DSA 和 DSAN 算法的寻优能力较差。采用 LSGA 框架的 DSA 算法，虽然提高了 DSA 算法的局部搜索性能，但其对 DSA 算法的性能提升有限，相比 LSGA-DSA 算法，RDMAD 算法具有更好的收敛性能。另外 ACO_DCOP 和 AED 算法都利用了种群对问题进行寻优，但从图中可以发现，RDMAD 算法在收敛和寻优质量上比这两种算法表现更好。在 EXP-2 上，由于测试问题密度过高，agent 间的约束增多，因此算法的性能受到一定影响，但 RDMAD 算法仍能保持较好的寻优性能，其收敛和寻优质量都优于 ACO_DCOP，相比 ACO_DCOP 提高了约 1.4%。同时 RDMAD 算法能够得到与 LSGA-DSA 和 AED 算法同等质量的解，且收敛优势明显，相比其他算法提高了约 1.1%～2.7%。同样的，在 EXP-3 和 EXP-4 上 RDMAD 算法优于 ACO_DCOP 约 4.4%和 2.8%，相比其他算法分别提高约 4.1%～17.2%和 0.6%～8.3%。可以看出，RDMAD 算法在无尺度网络问题上(EXP-3, EXP-4)的表现更优秀，收敛速度也具有明显优势。这表明 RDMAD 算法在求解结构化问题时性能显著。最后，在 EXP-5 测试问题上，RDMAD 算法同样表现出了良好的收敛和寻优性能，优于 ACO_DCOP 约 18.4%，同时相比其他算法提高了约 18.7%～62.7%。

表 2　每个问题在 20 个实例上的统计结果

Tab. 2　Results on 20 instances of each question

| 问题 | Mean±Std Dev | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RDMAD | ACO_DCOP | AED | DSA | LSGA-DSA | PDS-DSA | GDBA | DSAN |
| EXP-1 | **5340+79.7** | 5575+82.0 | 5463+72.6 | 5991+98.4 | 5431+71.0 | 5411+82.2 | 5797+86.4 | 6127+81.7 |
| + | | 20 | 18 | 20 | 17 | 16 | 20 | 20 |
| - | | 0 | 2 | 0 | 3 | 4 | 0 | 0 |
| *p-value* | | **0.000** | **0.001** | **0.000** | **0.001** | **0.007** | **0.000** | **0.000** |
| EXP-2 | 55439+210.6 | 56212+205.8 | **55400+225.0** | 56524+188.3 | 55456+211.5 | 56057+199.7 | 56129+149.9 | 57002+187.6 |
| + | | 20 | 10 | 20 | 9 | 20 | 20 | 20 |
| - | | 0 | 10 | 0 | 11 | 0 | 0 | 0 |
| *p-value* | | **0.000** | 0.601 | **0.000** | 0.709 | **0.000** | **0.000** | **0.000** |
| EXP-3 | **3648+81.9** | 3816+69.3 | 3853+61.8 | 4360+76.9 | 3845+77.1 | 3804+80.4 | 4137+75.6 | 4408+61.9 |
| + | | 20 | 18 | 20 | 19 | 17 | 20 | 20 |
| - | | 0 | 2 | 0 | 1 | 3 | 0 | 0 |
| *p-value* | | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |
| EXP-4 | **12334+92.0** | 12684+126.5 | 12424+101.9 | 13160+139.2 | 12413+90.3 | 12589+106.3 | 12935+101.1 | 13449+73.2 |
| + | | 20 | 17 | 20 | 15 | 20 | 20 | 20 |
| - | | 0 | 3 | 0 | 5 | 0 | 0 | 0 |
| *p-value* | | **0.000** | **0.010** | **0.000** | **0.025** | **0.000** | **0.000** | **0.000** |
| EXP-5 | **279+29.3** | 342+36.4 | 343+31.9 | 747+45.7 | 396+31.5 | 372+31.7 | 476+41.1 | 693+44.7 |
| + | | 20 | 17 | 20 | 20 | 19 | 20 | 20 |
| - | | 0 | 3 | 0 | 0 | 1 | 0 | 0 |
| *p-value* | | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |

(a) EXP-1 上的实验对比结果

(b) EXP-2 上的实验对比结果

(c) EXP-3 上的实验对比结果

(d) EXP-4 上的实验对比结果

(e) EXP-5 上的实验对比结果

图 4　各算法在不同测试问题上的收敛曲线

Fig. 4　Convergence curves of various algorithms on different test problem

## 4 结束语

有效利用群体智能求解分布式约束优化问题，是提高 DCOP 求解算法性能的新思路，本文提出了一种基于多种群的随机扰动蚁群算法求解分布式约束优化问题。该算法首先充分地利用种群特性，通过多种群分工配合分级更新策略更好地平衡算法的探索和开发能力，提高了算法的收敛速度和求解质量。在此基础上利用随机扰动策略增加种群多样性，避免算法陷入局部最优。将 RDMAD 算法与 ACO_DCOP 算法，以及其他 6 种目前最先进的非完备算法在三类基准问题上进行比较，RDMAD 算法在求解质量和收敛速度方面优势显著，且具有良好的稳定性。但是 RDMAD 缺少对 agent 局部收敛状态的利用，在以后的工作中将考虑引入信息熵等状态评价指标与随机扰动机制结合。

## 参考文献：


[1] 刘鸿福, 陈璟, 沈林成. 分布式约束满足问题及其在 MAS 任务分配中的应用 [J]. 计算机应用研究, 2009, 26 (2): 515-517. (Liu Hongfu, Chen Jing, Shen Lincheng. Distributed constraint satisfaction problem and its application to multi-agent system task allocation [J]. Computer Application Research, 2009, 26 (2): 515-517.)

[2] Leite R, Fabricio E, Barthes J. Distributed Constraint Optimization Problems: Review and perspectives [J]. Expert Systems with Applications, 2014, 41 (11): 5139-5157.

[3] Zhang W, Wang G, Xing Z, *et al.* Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks [J]. Artificial Intelligence, 2005, 161 (1-2): 55-87.

[4] Sultanik E A, Modi P J, Regli W C. On modeling multiagent task scheduling as a distributed constraint optimization problem [C]. In Proc. of the 20th IJCAI, 2007: 1531-1536.

[5] Hirayama K, Yokoo M. Distributed partial constraint satisfaction problem [C]. Principles and Practice of Constraint Programming-CP97. Springer Berlin Heidelberg, 1997: 222-236

[6] Gershman A, Meisels A, Zivan R. Asynchronous forward bounding for distributed COPs [J]. Journal of Artificial Intelligence Research, 2009, 34 (1): 61-88.

[7] Modi P J, Shen W M, Tambe M, *et al.* ADOPT: Asynchronous distributed constraint optimization with quality guarantees [J]. Artificial Intelligence, 2005, 161 (1): 149-180.

[8] Yeoh W, Felner A, Koenig S. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm [J]. Journal of Artificial Intelligence Research, 2010, 38: 85-133.

[9] Petcu A, Faltings B. A scalable method for multiagent constraint optimization [C]. IJCAI, 2005, 5: 266-271.

[10] Petcu A, Faltings B. Mb-dpop: A new memory-bounded algorithm for distributed optimization [C]. In Proceedings of the twentieth international joint conference on artificial intelligence, 2007: 1452-1457.

[11] Chen Z, Zhang W, Deng Y, *et al.* RMB-DPOP: Refining MB-DPOP by Reducing Redundant Inference [C]. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020: 249-257.

[12] Rashik, M. , Rahman, M. M. , Khan, M. M. *et al.* Speeding up distributed pseudo-tree optimization procedures with cross edge consistency to solve DCOPs [J]. Appl Intell, 2021, 51: 1733–1746.

[13] Okamoto, Zivan, Nahon. Distributed breakout: Beyond satisfaction [C]. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016: 447-453.

[14] Zivan R. Anytime local search for distributed constraint optimization [C]. International Joint Conference on Autonomous Agents and Multiagent Systems. DBLP, 2008: 1449-1452.

[15] Chen Z, Yu Z, He J, *et al.* A partial decision scheme for local search algorithms for distributed constraint optimization problems [C]. In Proceedings of the 16th international conference on autonomous agents and multiagent systems, 2017: 187-194.

[16] Chen Z, Liu L, He J. *et al.* A genetic algorithm based framework for local search algorithms for distributed constraint optimization problems [J]. Auton Agent Multi-Agent Syst, 2020, 34: 41.

[17] Farinelli A, Rogers A, Petcu A, *et al.* Decentralised coordination of low-power embedded devices using the max-sum algorithm [C]. In Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) , Estoril, Portugal, 2008: 639-646.

[18] Zivan R, Peled H. Max/min-sum distributed constraint optimization through value propagation on an alternating DAG [C]. International Conference on Autonomous Agents and Multiagent Systems. 2012: 265-272.

[19] Ottens B, Dimitrakakis C, Faltings B. Duct: An upper confidence bound approach to distributed constraint optimization problems [C]. In Proceedings of the 26th conference on Artificial Intelligence (AAAI) , Toronto, Canada, 2012: 528-533.

[20] Mahmud S, Choudhury M, Khan M M, *et al.* AED: An anytime evolutionary DCOP algorithm [C]. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020.

[21] Chen Z, Wu T, Deng Y, *et al.* An ant-based algorithm to solve distributed constraint optimization problems [C]. In Proc. of the 32th AAAI conference on Artificial Intelligence, 2018: 4654-4661.

[22] 薛宏全, 魏生民, 张鹏, 等. 基于多种群蚁群算法的柔性作业车间调度研究 [J]. 计算机工程与应用, 2013, 49 (24): 243-248. (Xue Hongquan, Wei Shengmin, Zhang Peng, *et al.* Flexible job-shop scheduling based on multiple ant colony algorithm [J]. Computer Engineering and Applications, 2013, 49 (24): 243-248.)

[23] 朱佑滔, 何志琴, 施文烨. 多种群蚁群算法在机械手臂路径规划中的设计与应用 [J]. 机械传动, 2021, 4: 160-165. (Zhu Youtao, He Zhiqin, Shi Wenye. Design and Application of Multi-colony Ant Algorithm in Path Planning of Manipulator [J]. Journal of Mechanical Transmission, 2021, 4: 160-165.)

[24] 陈佳, 游晓明, 刘升, 等. 结合信息熵的多种群博弈蚁群算法 [J]. 计算机工程与应用, 2019, 55 (16): 170-178. (Chen Jia, You Xiaoming, Liu Sheng, *et al.* Entropy-game based multi-population ant colony optimization [J]. Computer Engineering and Applications, 2019, 55 (16): 170-178.)

[25] Zivan R, Okamoto S, Peled H. Explorative anytime local search for distributed constraint optimization [J]. Artificial Intelligence, 2014, 212: 1-26.

[26] Ziyu Chen, Zhen He, Chen He. An improved DPOP algorithm based on breadth first search pseudo-tree for distributed constraint optimization [J]. Applied Intelligence, 2017, 47: 607-623.

[27] Zivan R, Okamoto S, Peled H. Explorative anytime local search for distributed constraint optimization [J]. Artificial Intelligence, 2014, 212: 1-26.

[28] Barabási, A-L, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286: 509-512.

[29] Arshad M, Silaghi M C. Distributed simulated annealing [J]. Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems, 2004, 112.